On Identifying and Managing Relationships in Multi-Agent Systems

Ronald Ashri and **Michael Luck**
University of Southampton, Southampton, SO17 1BJ, United Kingdom
{ra00r,mml}@ecs.soton.ac.uk

Mark d'Inverno
University of Westminister, London, E1 4NS, United Kingdom
dinverm@westminster.ac.uk

Abstract

Multi-agent systems result from interactions between individual agents. Through these interactions different kinds of relationships are formed, which can impact substantially on the overall system performance. However, the behaviour of agents cannot always be anticipated, especially when dealing with open and complex systems. Open agent systems must incorporate *relationship management* mechanisms to constrain agent actions and allow only desirable interactions. In consequence, in this paper we tackle two important issues. Firstly, in addressing *management*, we identify the range of different control mechanisms that are required and when they should be applied. Secondly, in addressing *relationships*, we present a model for identifying and characterising relationships in a manner that is application-neutral and amenable to automation.

1 Introduction

Agent-based computing is often presented as a viable paradigm for tackling problems where the main challenge is the need to enable dynamic interactions between heterogeneous components, in which each component may have its own thread of control [Jennings, 2000]. Agents are understood as independent entities, capable of individual action and working towards their design goals and, as such, represent a natural, intuitive approach to these problems. Furthermore, agents are also *social* entities that interact by performing actions that affect their environment and other agents in the environment. Such actions can enable coordination (in which agents arrange their individual activities in a coherent manner), collaboration (in which agents work together to achieve a common objective) or competition (in which agents contend for access to common resources), and so on. Through these interactions, different kinds of relationships are formed between agents, and they can impact substantially on the overall system functioning.

The challenge for the system designer is to ensure that only the *right* kinds of relationships develop so that the overall system functions within acceptable parameters. It is not enough for a designer to assume that the only interactions between agents that will take place are those explicitly specified at design-time, especially when dealing with *autonomous* entities operating in open environments where agents may join or leave the system at any time and no assumptions are made about agent behaviour. As a result, open agent systems must incorporate mechanisms, beyond the design of individual agents, to constrain agent actions and allow only desirable interactions. We refer to these as *relationship management* mechanisms.

To address the problem of relationship management, we must be aware of the different types of management that may be required in a multi-agent system and also be able to identify and characterise relationships in order to understand what type of management is best suited for the situation at hand. It is exactly these concerns that we attempt to tackle in this paper. This paper advances the current understanding of how to deal with the issue of relationship management in two important ways. Firstly, in addressing *management*, we identify the range of different control mechanisms that are required and when they should be applied. Secondly, in addressing *relationships*, we present a model for identifying and characterising relationships in a manner that is application-neutral and amenable to automation.

We begin by presenting a motivating scenario that outlines some realistic situations that require relationship management mechanisms. Subsequently, we identify situations in which mechanisms for managing relationships can be applied and we define a space of possible management mechanisms types. We then focus on the issue of relationship identification through an analysis of individual agent actions and perceptions. Finally, we conclude with a brief look at related work and outline the way in which the contributions of this paper add to such work.

2 Motivating Scenario

Let us consider a typical computer science research lab, where communication and cooperation between researchers is facilitated by an application developed as a multi-agent system. Each researcher is represented by an agent that makes public to the lab's network, through the researcher's personal computer, their personal profile (interests, publications, availability) as well as research material (downloaded papers, presentations, software, links to online material) that they have stored locally. The application also supports researchers

while working away from their desks through wireless connections on laptops or more limited personal digital assistants. Finally, visitors to the lab are provided with similar functionality, through mobile devices, so that they can more easily locate people within the lab.

For such an application to be successful we must, at the very least, ensure that agents make available the right kinds of information about the researcher they represent, so as to add to the overall system functioning, and that agents do not abuse the system, causing degradation of the abilities of others, e.g. by making too many queries and using up bandwidth. Such overarching system goals can be achieved by the appropriate management of relationships between agents. Below we present some situations that call for relationship management.

1. When retrieving files from a researcher's local hard disk limits on how much can be retrieved, and how many agents can simultaneously retrieve at any given moment should be set. This would ensure that no single machine suffers from too much demand.
2. A researcher may want to tailor access to locally stored research material based on who is accessing it. For example, those working on common projects might access work in progress, while PhD students might not have access to all material of their supervisor.
3. Each agent should follow certain conventions. A researcher should not restrict access to their own material while expecting to have full access to others.
4. Groups of researchers in the lab could benefit from closer relationships due to a common interest in a specific subject or project. This tighter relationship could be reflected by the creation of *special interest groups* where the sharing of materials between them takes place at a more frequent rate and without explicit requests to receive them.
5. Different devices offer different capabilities. A document may not be retrieved by a a visitor's PDA if it is not capable of displaying it, but the PDA may request a commitment to send the document to the visitor via e-mail.

The examples illustrate the different types of problems and the different needs relating to the management of relationships between agents. Even though some may seem to be amenable to traditional file management techniques (e.g. the first two) the *range* of situations described is more complex and the envisaged application of regulation is more dynamic and fine-grained, and is handled by the agent system and not a human administrator.

The examples can be roughly divided into those that may demand the establishment of compulsory, rigid control of relationships and those where control could be more flexible or even optional. The first two examples could fall in the former category and the last three in the latter. Furthermore, some define the need for control based on the current *situation* (Example 1), while others view it based on *who* is interacting with whom (Example 2) or past history of interactions (Examples 3 and 4). Agents can benefit from the establishment of long term commitments between a group of agents, such as forming interest groups (Example 4) or short term commitments for the fulfilment of a specific goal, such as dealing with access through limited capability devices (Example 5). Finally, the needs of the application may call for agents to adopt a specific stance due to their individual characteristics, such as in Example 2 and Example 5 where we have supervisors and supervisees, and local and visiting researchers respectively.

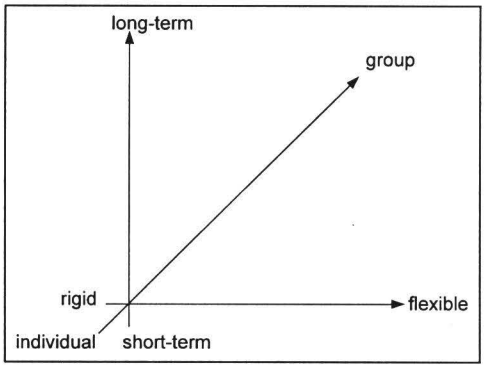

Figure 1: Space of management types

3 Relationship Management Types

These different cases provide a guide for a more comprehensive categorisation of situations that may *activate* the need for management and the *kind* of management that can be applied.

We identify three different criteria that may activate the need for management. Management of relationships can be required when a specific event occurs (e.g. an agent accessing a researcher's profile), when the environment corresponds to a predefined situation (e.g. the number of agents simultaneously accessing a profile exceeds four) or when a specific agent is performing an action (e.g. a student accessing a supervisor's profile). Thus we can divide the cases into *event-based*, *situation-based* and *agent-attribute-based*. These cases are not mutually exclusive but can occur at the same time. For example, management may be needed for an agent with specific attributes in a particular situation when a specific event occurs.

The kind of management can be defined along three different dimensions: firstly, the *number of agents* it refers to; secondly, the *rigidity* of the applied regulation, which indicates to which extent agents can choose not to follow the regulation; and finally, the *duration* of management, which indicates for how long agents should follow the applied regulation. Using these dimensions we can define a *management space*, illustrated in Figure 1. By using the notion of a *space*, rather than rigid categories, we want to emphasise that the understanding of relationship management needs to better reflect the *range* of situations that may occur. So, for example, management may not need to be simply rigid or optional but also the whole range of flexibility in between.

With knowledge of when management may be applied and what form it can take we can more clearly characterise and compare different management methods. For example, re-

turning to our scenario, we could control the level of access to material (Example 1) via a regulation that agents could choose to obey, which is activated when the agent performs the action of accessing the information (i.e. it is event based) and applies only to that agent accessing information (i.e. it refers to only one agent). Alternatively, the same problem could be tackled through a long-term regulation that applies to all agents and cannot be questioned. Each method has its own pros and cons and can be characterised through the concepts we provide.

4 Identifying Relationships

However, before one can manage relationships one must be able to identify and characterise them. We begin by briefly outlining some foundational concepts that we will use to formalise the model later on. To do so, we adopt the Z specification language [Spivey, 1992]. Based on the SMART framework [d'Inverno and Luck, 2001], we start by defining two primitives, *attribute* and *action*. Formally, these primitives are specified as given sets which means that we say nothing about how they might be represented for any particular system. In addition, two secondary concepts, *goal* and *environment*, are specified in terms of attributes. Attributes are simply features of the world, and are the only characteristics that are manifest. Actions can change environments by adding or removing attributes.

$$[Attribute, Action]$$

Goals, in this context, are desirable environmental states described by non-empty sets of attributes. An environment is a set of attributes that describes all the features within that environment.

$$Environment == \mathbb{P}_1\, Attribute$$
$$Goal == \mathbb{P}_1\, Attribute$$

Using these primitive concepts we can describe an agent as shown in the following schema. An agent is described by a set of attributes, can perform certain actions and has a number of goals to achieve.

```
┌─ Agent ──────────────────────
│ attributes : ℙ Attribute
│ actions : ℙ Action
│ goals : ℙ Goal
└──────────────────────────────
```

The model for relationship identification builds on just these concepts and can deal with a wide variety of situations.

4.1 Influence Types

As discussed above, relationships can take a number of forms ranging from cooperation towards a common goal to competition for possession of, or access to, a common resource. When such relationships occur they may affect the actions agents perform or were intending to perform. In such situations, the ability of one agent to achieve its goals becomes *dependent* on another agent's actions. It is these dependencies between agents that we are categorising through an analysis of the agent's respective actions and perceptions of the environment.

We begin by defining the state of an agent as those aspects of the environment it can perceive, along with its actual situation in the environment.

```
┌─ AgentState ─────────────────
│ Agent
│ possiblepercepts : ℙ Attribute
│ situation : ℙ Attribute
└──────────────────────────────
```

The state of the entire multi-agent system would then be given by the environment and the states of each individual agent in that environment.

```
┌─ MAState ────────────────────
│ environment : ℙ Attribute
│ agents : ℙ AgentState
├──────────────────────────────
│ ∀ a : agents • a.attributes ⊂ environment ∧
│          a.situation ⊂ environment
│ ∀ a : agents • a.possiblepercepts ⊆
│          environment
└──────────────────────────────
```

Now, if we consider that an agent usually takes into account some measure of the current state of the environment and performs actions based on those measures which, in turn, may change the environment, we have an indication of which aspects of an agent we should investigate to understand under which conditions two agents may be related. We must examine the overlaps between the aspects of the environment that one agent can view, through its *sensing* capabilities, and those aspects it can affect, through its *actuator* capabilities, in relation to other agents. We call the former the *viewable environment* and the latter the *region of influence*. The *Viewable Environment* schema formalises the former notion.

```
┌─ ViewableEnvironment ────────
│ MAState
│ viewable : AgentState ⇸ ℙ Attribute
├──────────────────────────────
│ dom viewable = agents
│ ∀ a : agents • viewable a ⊆ environment
└──────────────────────────────
```

An analogous schema describes the region of influence.

```
┌─ RegionOfInfluence ──────────
│ MAState
│ regioninfluence : AgentState ⇸ ℙ Attribute
├──────────────────────────────
│ dom regioninfluence = agents
│ ∀ a : agents • regioninfluence a ⊆ environment
└──────────────────────────────
```

In Figure 2, we illustrate these concepts. The regions of the environment that agents view and take into account are represented as ellipses while the regions they are able to affect, i.e. the regions of influence, are represented as pentagons. Given this information we can infer that Agent A and Agent B could be related, with A able to directly affect the viewable

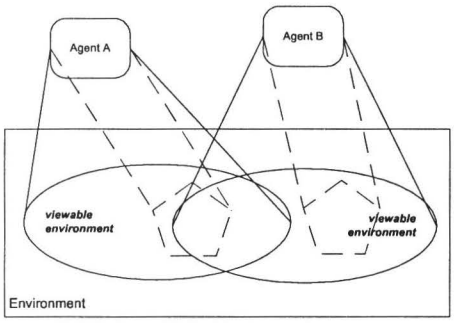

Figure 2: Region of influence affects viewable environment

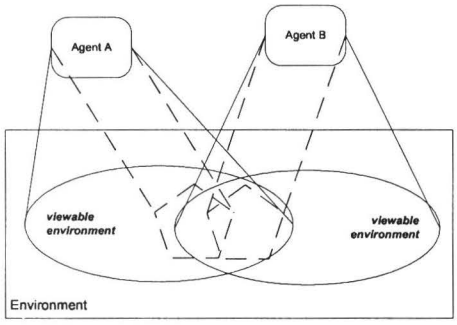

Figure 3: Regions of influence overlap

environment of B since it partly falls under A's region of influence. In other words, B can be *influenced* by the actions of A. Agent A, on the other hand, can in no way be influenced by B. Crucially, A cannot *directly* affect the results of an action of B because it has no influence in the region of influence of B. Returning to our scenario, such a situation could occur if the overlap between the viewable environments represented the documents stored in A's desktop computer. Agent B, with the task of reporting to other agents on all documents of a specific type (e.g. papers on multi-agent systems), could periodically view the documents stored by A (i.e. sample the environment) waiting for a relevant document to appear before informing other agents about its existence. So, whenever A performs an action that adds a relevant document to its public document store, it will eventually *influence* B's actions, since B must now inform interested parties about this addition.

In Figure 3, the situation is one where the regions of influence overlap. This means that both agents can have a direct impact on the actions of each other. Thus, an action from either agent could affect the environment in such a way that a goal of the other agent is upset. Returning to our scenario, this could happen if the two agents were both attempting to retrieve a document from a public document store that sets a limit to the number of documents retrieved.

At this point, we should clarify that although in our diagrams the regions of influence have always fallen under the viewable environment of the respective agents this is only for illustrative reasons. In fact, there is no requirement for the viewable environment and the region of influence of an agent to overlap at all. If the region of influence of an agent does not fall under that agent's viewable environment that agent would not be able to view the results of its actions, a situation that is not improbable. Even humans often do not fully realise the ramifications of their actions. The more usual case is when only part the region of influence of an agent falls under the viewable environment. In other words, the agent may not be fully aware of all the implications of its actions.

Now, through the concepts discussed so far we can outline four different ways in which one agent can influence another.

Weak influence A *weak influence* relationship occurs when an agent is able to affect aspects of the environment that another agent uses to decide what actions to perform (i.e. aspects of the environment the agent can perceive). Although an influence relationship can lead to a different outcome for the influenced agent's goal it cannot directly affect actions of that agent.

Agent B is *weakly influenced* by Agent A if and only if (i) the intersection of A's region of influence and B's viewable environment is non-empty, and (ii) the intersection of A's region of influence and B's region of influence is empty.

$$
\begin{array}{l}
\textit{WeakInfluence} \\
\hline
\textit{RegionOfInfluence} \\
\textit{ViewableEnvironment} \\
\textit{weakinfluenced} : \textit{AgentState} \leftrightarrow \textit{AgentState} \\
\hline
\forall a, b : \textit{AgentState} \mid a \neq b \bullet \\
\quad (b, a) \in \textit{weakinfluenced} \Leftrightarrow \\
\quad\quad \textit{regioninfluence}\ b \cap \textit{regioninfluence}\ a = \{\} \wedge \\
\quad\quad \textit{viewable}\ b \cap \textit{regioninfluence}\ a \neq \{\} \\
\hline
\end{array}
$$

Strong influence A *strong influence* relationship occurs when an agent is able to affect both the viewable environment of another agent as well as its region of influence. In this case an agent can directly affect the goals of another agent because it can act on exactly those aspects of the environment that may represent desirable environmental states for the other agent.

Agent B is *strongly influenced* by Agent A if and only if (i) the intersection of A's region of influence and B's viewable environment is non-empty, and (ii) the intersection of A's region of influence and B's region of influence is non-empty.

$$
\begin{array}{l}
\textit{StrongInfluence} \\
\hline
\textit{RegionOfInfluence} \\
\textit{ViewableEnvironment} \\
\textit{stronginfluenced} : \textit{AgentState} \leftrightarrow \textit{AgentState} \\
\hline
\forall a, b : \textit{AgentState} \mid a \neq b \bullet \\
\quad (b, a) \in \textit{stronginfluenced} \Leftrightarrow \\
\quad\quad \textit{regioninfluence}\ b \cap \textit{regioninfluence}\ a \neq \{\} \wedge \\
\quad\quad \textit{viewable}\ b \cap \textit{regioninfluence}\ a \neq \{\} \\
\hline
\end{array}
$$

Sneaky influence A sneaky influence relationship occurs when an agent is able to affect the region of influence of another agent but not the viewable environment. This, of course, implies that the influenced agent cannot view the results of its actions, so cannot be aware that some other agent is affecting those results.

Agent B is *sneakily influenced* by Agent A if and only if (i) the intersection of A's region of influence and B's viewable environment is empty, and (ii) the intersection of A's region of influence and B's region of influence is non-empty.

$$
\begin{array}{|l}
\hline
\textit{SneakyInfluence} \\
\textit{RegionOfInfluence} \\
\textit{ViewableEnvironment} \\
\textit{sneakyinfluenced} : \textit{AgentState} \leftrightarrow \textit{AgentState} \\
\hline
\forall a, b : \textit{AgentState} \mid a \neq b \bullet \\
\quad (b, a) \in \textit{sneakyinfluenced} \Leftrightarrow \\
\quad\quad \textit{regioninfluence}\ b \cap \textit{regioninfluence}\ a \neq \{\} \wedge \\
\quad\quad \textit{viewable}\ b \cap \textit{regioninfluence}\ a = \{\} \\
\hline
\end{array}
$$

No influence Finally, when an agent cannot affect the viewable environment or the region of influence of another agent, no direct relationship can develop between them.

Agent B is *not influenced* by Agent A if and only if (i) the intersection of A's region of influence and B's viewable environment is empty, and (ii) the intersection of A's region of influence and B's region of influence is empty.

$$
\begin{array}{|l}
\hline
\textit{NoInfluence} \\
\textit{RegionOfInfluence} \\
\textit{ViewableEnvironment} \\
\textit{notinfluenced} : \textit{AgentState} \leftrightarrow \textit{AgentState} \\
\hline
\forall a, b : \textit{AgentState} \mid a \neq b \bullet \\
\quad (b, a) \in \textit{notinfluenced} \Leftrightarrow \\
\quad\quad \textit{regioninfluence}\ b \cap \textit{regioninfluence}\ a = \{\} \wedge \\
\quad\quad \textit{viewable}\ b \cap \textit{regioninfluence}\ a = \{\} \\
\hline
\end{array}
$$

These four types of influence can now act as a guide to characterise a range of specific kinds of relationships. For example, a competitive relationship for access to common resources can only take place if both agents can strongly influence each other, i.e. their regions of influence and viewable environments overlap. A supervisor-student relationship is one where the supervisor can strongly influence the student (e.g. by providing direct guidance on what research the student should do) and the student can weakly influence the supervisor (e.g. by coming up with new results that may convince the supervisor to change research direction).

4.2 Effect of influence on actions and goals

In order to have a clearer understanding on exactly how one agent could affect the goals or actions of another we provide an analysis of the different cases. This is based on two assumptions: firstly, that agents perform actions because they want to achieve goals; and secondly, that goals agents are trying to achieve fall under their region of influence. As such, it makes sense to define the relationships that evolve through the interactions between agents in terms of the contribution that such interactions have towards the achievement of their goals. However, if an agent's goal cannot be achieved within that agent's region of influence the agent must seek assistance from another agent that has access to the region of the environment within which the goal can be achieved. In this paper we do not look at those situations.

Weak influence relationships

When only weak influence relationships occur, the influencing agent cannot directly impact goals. Nevertheless, it can still have a significant effect on the way the influenced agent achieves a goal, or whether the goal can be achieved at all. In essence, an agent could either be influenced so as to change its actions in order to achieve a goal or to change the goal completely. Below, we outline the different scenarios.

Goal does not change In the first type of case, the goal of the agent does not change as a result of the influencing agent. However, the actions performed to achieve the goal might change, as might the exact results of the actions, because of the goal.

No effect The influencing agent has no impact on the outcome of the goal because the attributes of the environment that are affected by the influencing agent are not taken into account for the execution of an action by the influenced agent.

Outcome of action changes Here the influencing agent affects the environment in such a way that the outcome of the action performed by the influenced agent changes. However, the *goal* of the influenced agent does not change. For example, consider an agent with the goal of compiling a list of all researchers with an interest in the subject of argumentation. The goal is satisfied as long as such a list exists. The agent compiles the list by asking other agents to declare their interest or not in the subject. The queried agents influence the *outcome* of the action by providing an answer. In any case, the goal is eventually achieved. However, the exact values described in the list have been influenced by others.

Action changes Agents may influence another agent to such an extent that the agent needs to change its planned actions in order to achieve the goal. For example, if some agents refuse to declare whether they are interested in argumentation, the agent may need to follow an alternative route, such as looking at their list of publications for evidence of an interest in the subject.

Goal changes The second type of scenario is when the influencing agent may change the environment in such a way that the influenced agent has to change its goal entirely. For example, let us assume that Agent A has two goals. The first goal, of primary importance, is to discover any paper on negotiation, and the second goal, of secondary importance, is to discover papers relating to middleware. If A was pursuing the secondary goal and discovers that new papers relating to the primary goal have been posted by B, A must then change goals to reflect the change in the environment. As such, B has sufficiently influenced A, through actions that impacted on just the agent's viewable environment, so that A changed its goal.

Strong and sneaky influence

Strong and sneaky influence relationships can impact on a goal in a more immediate way since agents could change exactly those attributes that represent a goal state for another agent. We identify three main cases below.

No change In the first case, the actions of Agent A do not affect the goal of Agent B. This means that although A is able to act in the region of influence of B, it does not perform actions that upset goal states for B.

Goal upset An agent can perform an action that changes the environment in such a way that a goal state of another agent is upset. For example, one agent may wish to access a document but cannot because another agent is already accessing it or has placed restrictions on its access.

Goal aided Alternatively, an agent can perform an action that helps towards creating the goal state of another agent. Such actions may have been intentional or may occur unintentionally. For example, if an agent has the goal of discovering a paper on auctions and another posts that paper, it inadvertentently aids the second one in achieving its goal.

5 Conclusions

The issue of agent relationship management has been tackled in a variety of settings through existing research. In the distributed systems area there has been work on policy specifications. Policies are understood as *"one aspect of information which influences the behaviour of objects within the system"* [Sloman, 1994] and deal with the definition of rules that identify what actions can or cannot be performed with respect to a target. Although there is a wide body of work on policy specification (e.g. [Barker, 2000; Damianou *et al.*, 2001]), only recently has such work been applied to multi-agent systems [Dulay *et al.*, 2001]. Closer to the field of multi-agent systems is research into the issues of interaction between agents with a view of such interdependencies as social structures [Castelfranchi, 1990], which has led to the adoption of the notion of *norms*. A norm is, similar to a policy, considered as a means of regulating behaviour between agents. Various ways of representing norms have developed, such as obligations, authorisations and conventions [Dignum, 1999], commitments [Jennings, 1993] and mental attitudes [Conte and Castelfranchi, 1995], as well as methods for reasoning about norms (e.g. [Conte *et al.*, 1999; y Lopez *et al.*, 2002]).

However, existing research has not addressed the need for identifying the whole range of relationship management structures that can be developed (usually looking at just the case of optional or compulsory rules) and has not provided appropriate models for identifying exactly what kinds of relationships develop in a manner that can be widely applied and incorporated into practical application settings.

In this paper we provide the groundwork for addressing these issues, which are central to the effective design of multi-agent systems. Firstly, through our description of the space of management mechanisms we have a single basis by which to access and compare the various frameworks developed so far. Secondly, through the conceptual model for the identification of relationships we have a means to rationalise the process of when and how relationships should be managed and include it within a design methodology for multi-agent systems. Further work will move in two directions. On the one hand, we will move towards a relationship management framework, based on regulations, that covers the range of management types presented here and place it within a wider solution for multi-agent systems infrastructure [Ashri *et al.*, 2002]. On the other hand, we will proceed with the characterisation of agent relationships, looking at issues such as how regions of influence are affected when one agent makes a commitment to perform an action for another agent and where *groups* of agents are involved.

References

[Ashri *et al.*, 2002] R. Ashri, M. Luck, and M. d'Inverno. Infrastructure Support for Agent-based Development. In M. d'Inverno, M. Luck, M. Fisher, and C. Preist, editors, *Foundations and Applications of Multi-Agent Systems*, volume 2403 of *LNAI*, pages 73–88. Springer, 2002.

[Barker, 2000] S. Barker. Data protection by logic programming. In J. Loyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamedessi, L.M. Pereira, Y.Sagiv, and P.J. Stuckey, editors, *Computational Logic-CL2000*, volume 1861 of *LNCS*, pages 1300–1315. Springer, 2000.

[Castelfranchi, 1990] C. Castelfranchi. Social power. In Y. Demazeau and J.-P. Muller, editors, *Decentralized AI*. Elsevier, 1990.

[Conte and Castelfranchi, 1995] R. Conte and C. Castelfranchi. Norms as mental objects: From normative beliefs to normative goals. In C. Castelfranchi and J. P. Muller, editors, *From Reaction To Cognition*, volume 957 of *LNAI*. Springer, 1995.

[Conte *et al.*, 1999] R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm-acceptance. In J. Muller, M. Singh, and A. Rao, editors, *Intelligent Agents V (ATAL98)*, volume 1555 of *LNAI*, pages 319–333. Springer, 1999.

[Damianou *et al.*, 2001] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Specification Language. In *Workshop on Policies for Distributed Systems and Networks (Policy2001)*, volume 1995 of *LNCS*, pages 18–39. Springer, 2001.

[Dignum, 1999] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, (7):69–79, 1999.

[d'Inverno and Luck, 2001] M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer, 2001.

[Dulay *et al.*, 2001] N. Dulay, N. Damianou, E. Lupu, and M. Sloman. A policy language for the management of distributed agents. In M. J. Wooldridge, G. Weiss, and P. Cinacarini, editors, *Agent-Oriented Software Engineering II*, volume 2222 of *LNAI*, pages 84–100. Springer, 2001.

[Jennings, 1993] N. Jennings. Commitments and conventions: The foundation of co-ordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[Jennings, 2000] N. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.

[Sloman, 1994] M. Sloman. Policy driven management for distributed systems. *Network and Systems Management*, 2(4):333–360, 1994.

[Spivey, 1992] J.M. Spivey. *The Z Notation*. Prentice Hall, 2nd edition, 1992.

[y Lopez *et al.*, 2002] F. Lopez y Lopez, M. Luck, and M. d'Inverno. Constraining Autonomy through Norms. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 674–681. ACM Press, 2002.